NASA/TM—2014-216621

Development of an Object-Oriented Turbomachinery Analysis Code Within the NPSS Framework

Scott M. Jones
Glenn Research Center, Cleveland, Ohio

January 2014

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at *http://www.sti.nasa.gov*

- E-mail your question to *help@sti.nasa.gov*

- Fax your question to the NASA STI Information Desk at 443–757–5803

- Phone the NASA STI Information Desk at 443–757–5802

- Write to:
  STI Information Desk
  NASA Center for AeroSpace Information
  7115 Standard Drive
  Hanover, MD 21076–1320

NASA/TM—2014-216621

Development of an Object-Oriented Turbomachinery Analysis Code Within the NPSS Framework

Scott M. Jones
Glenn Research Center, Cleveland, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

January 2014

This work was sponsored by the Fundamental Aeronautics Program at the NASA Glenn Research Center.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076–1320

National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Available electronically at http://www.sti.nasa.gov

Development of an Object-Oriented Turbomachinery Analysis Code Within the NPSS Framework

Scott M. Jones
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Abstract

This paper describes the development of a computational method to solve the equations of fluid flow across a turbomachinery blade row. The work was done within the Numerical Propulsion System Simulation (NPSS) computational framework and results in a collection of objects which use an implicit formulation of the turbomachinery equations to solve for the fluid exit conditions across each blade row. The object-oriented approach allows for easy development and evaluation of the empirical correlations traditionally used for turbomachinery loss, deviation, and blockage effects.

Introduction

The turbomachinery components in an aircraft gas turbine engine are the most critical to the overall system energy required to operate. Even with modern Computational Fluid Dynamics (CFD) techniques, the full 3-D fluid flowfield through the compressor and turbine components is currently impossible to quickly and reliably predict. Therefore, simpler 2-D methods have evolved to enable rapid estimation of the expected performance of these components; these methods rely largely on empirical knowledge and cascade tests to produce acceptable sensitivity to design inputs to narrow down the turbomachinery design parameters. This preliminary design can then be subjected to more rigorous CFD analysis. This paper discusses the development of an object-oriented code within the Numerical Propulsion System Simulation (NPSS) framework to perform the 2-D design and analysis of turbomachinery components.

Nomenclature

A	Fluid area
$\dot{m}$	Fluid massflow rate
h	Fluid enthalpy
MN	Fluid Mach number
p	Fluid (static) pressure
r	Radius of fluid element
s	Span or spanwise direction
T	Fluid (static) temperature
V	Fluid velocity
α	Swirl, angle between the fluid velocity vector and the meridional plane
β	Relative angle between the fluid velocity vector and the meridional plane
δ	Deviation, angle between the fluid exit relative angle and the blade exit angle
ρ	Fluid density
φ	Meridional angle, angle between the fluid meridional velocity and machine axis
ω	Blade row angular speed
$\bar{\omega}$	Non-dimensional total pressure loss parameter

Subscripts:

1	Property at station 1
2	Property at station 2
m	Meridional component
r	Radial component
t	Stagnation or total property
z	Axial component
i	Property of the i^{th} fluid element
θ	Tangential component

Background

With the possible exception of the combustor, turbomachinery components are the most complicated devices to design in a modern gas turbine engine. The 3-D flowfield within these components is extremely complex. Even neglecting the considerable computational time required, modern CFD techniques have yet to accurately predict the performance of such devices without being calibrated to specific known geometries and results. Because of this the turbomachinery designer must rely on alternative methods to quickly analyze and predict the performance of potential component designs. Such methods must be sufficiently accurate in order to prevent wasted effort on detailed design, CFD, and testing.

During the preliminary or conceptual design phase of an aircraft engine, the turbomachinery designer has a need to estimate the effects of a large number of design variables such as flow size, stage count, blade count, and radial position on the weight and efficiency of a turbomachine. Computer codes are invariably used to perform this task; however, such codes are often very old (in software terms) and written in outdated languages such as IBM's FORTRAN with arcane input files with the original author and code expert no longer available. Such codes are rarely adaptable to new architectures or unconventional layouts.

Given the need to perform these kinds of preliminary design trades, a modern 2-D turbomachinery design and analysis code has been written using the Numerical Propulsion System Simulation (NPSS) framework (Ref. 1). NPSS was chosen for several reasons: first, NPSS is a recognized and used state-of-the-art gas turbine engine performance analysis tool and likely to remain so for the foreseeable future. Adding a rapid 2-D turbomachinery design and analysis capability to NPSS is a logical fit. NPSS is an object-oriented code; as such, using it gives access to thermodynamics property packages, the `FlowStation` state object, and the `Solver` object immediately without having to code them separately. The other major advantage of object-oriented code is that, properly written, objects like loss models and deviation models can be easily updated or even switched as necessary.

OTAC Development Process

Development of the Object-oriented Turbomachinery Analysis Code (OTAC) started with a software requirements document identifying what the code is meant to do at a basic level. Without going into detail OTAC is being written to perform 2-D aerodynamic design and analysis of turbomachinery including compressors and turbines of both axial and centrifugal design. This sets the high-level physics of the problem that need to be modeled; based on assumptions about the flow a set of governing equations is developed. With the overall problem in mind the code architecture is then segregated into a set of intuitive "objects", each of which serves a specific purpose. For OTAC the existing NPSS `FlowStation` object was modified to represent the complete thermodynamic and velocity state of a given mass of fluid, including relative properties. Two important new objects are the `BladeSegment` object and the `BladeRow` object. The `BladeSegment` object represents the changes a fluid undergoes along a

streamline as it passes through a blade, and the `BladeRow` object represents a single blade row containing some number of flow streams. The existing NPSS `Solver` object was used as the means to solve the governing equations.

Equations Representing a Meanline Process

The physical relationships and associated equations used in OTAC follow from the software requirements and object set. Figure 1 shows a fluid element moving through an annulus of arbitrary radius. The state of the fluid element at any point is completely described by a set of seven attributes or properties:

- A massflow rate or mass flux through a given area normal to the meridional velocity
- Two thermodynamic state properties, such as pressure and temperature
- Three velocity state properties, such as velocity, tangential (swirl) angle, and meridional angle
- A location parameter, such as the fluid element radius

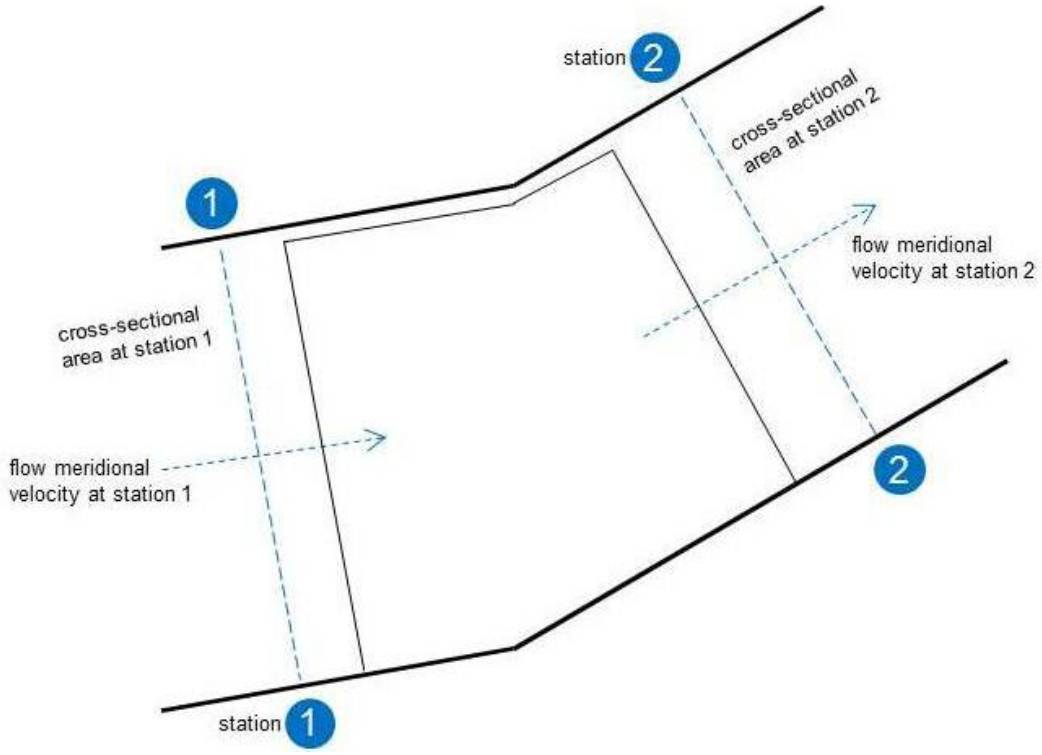

Figure 1.—Fluid flow through an annulus.

The fluid element begins at state 1, is acted upon by a device such as a blade row, and ends up at state 2. Given known fluid element properties at station 1, seven equations are required to uniquely determine its properties at station 2. The fluid element is assumed to undergo an adiabatic, steady state, circumferentially uniform process in an annulus where its angular momentum is changed by a blade row. The blade row rotational speed ω is known, although it may be zero (i.e., non-rotating). The following seven equations result:

$$\dot{m}_{m1} = \dot{m}_{m2} \tag{1}$$

$$h_{t2} - h_{t1} = \omega\left(r_2 V_{\theta 2} - r_1 V_{\theta 1}\right) \tag{2}$$

$$p_{t2} = p_{t2}\,\text{ideal} - f_{\text{loss}}(\text{args}) \tag{3}$$

$$\beta_2 = \text{blade metal exit angle} + \delta \tag{4}$$

$$r_2 = \text{machine radius at station 2} \tag{5}$$

$$A_{m2} = \text{machine area at station 2} \tag{6}$$

$$\phi_2 = \text{machine annulus angle at station 2} \tag{7}$$

Equation (1) is meridional continuity; it assumes the blade row can pass the given flow rate without choking. Equation (2) is the Euler turbomachinery equation. Equation (3) relates the fluid total pressure at station 2 to the ideal total pressure that would have resulted from an isentropic process minus some kind of loss; this loss function is an extremely complex relationship and will be discussed later. Equation (4) assumes the flow is turned such that it follows and exits primarily in the direction of the blade row with a minor adjustment δ equal to the angular deviation of the flow from the blade exit angle. By convention a positive value of deviation means the flow was not turned as much as it would have been if it had exited at the same angle as the blade. Estimation of deviation is done through an empirical relationship and will also be discussed later. Equations (5) to (7) represent the natural assumption that the fluid element is constrained by the machine and must have the same average radius and cross-sectional area. The flow meridional direction must also follow the machine average hub and casing angle. These equations assume that the flow is unseparated with negligible boundary layer at stations 1 and 2.

While a lot of assumptions seem to have been made, these equations are nevertheless sufficient to describe the process across nearly any type of turbomachine blade row: axial, centrifugal, mixed, radial, compression or expansion. However, as the fluid mass flow rate and area increase, there will be substantial change in various fluid properties in the spanwise direction which cannot be adequately accounted for using a single average condition.

Equations Representing a Streamline Process

When the spanwise fluid property variations are small or are assumed to be reflected in an average state, a meanline analysis is sufficient. More often, however, it is desired to account for these variations by dividing the flow into multiple regions as shown in Figure 2. This is called a spanline or streamline analysis. The equations for a streamline analysis are similar to the meanline equations presented above but with minor variations. Equations (1) to (4) are unchanged, except to note that the blade exit angle in Equation (4), a constant in meanline analysis, is now a function of distance along the span. Likewise the fluid meridional angle in Equation (7) is assumed to be a function of distance along the span. Equations (5) and (6) are altered under the following assumptions: first, the flow in aggregate must still be at the same

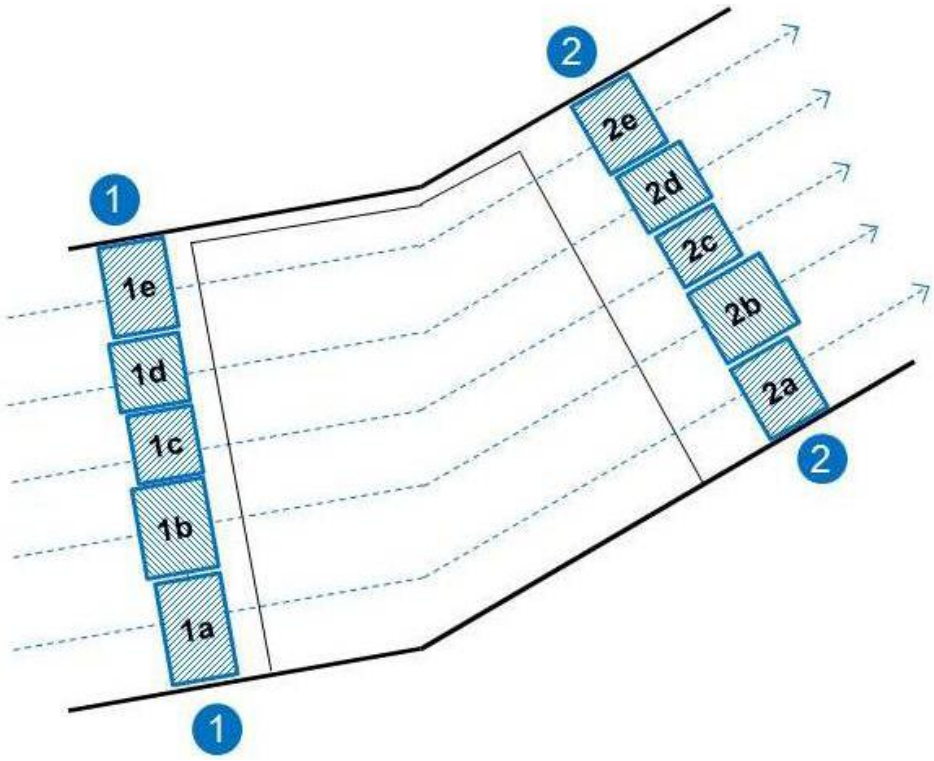

Figure 2.—Fluid segments for 5 streamlines.

average radius as the machine, and each individual fluid element radius must be such that the entire span or machine area is covered by the flow without gaps between the fluid elements. Also, by definition of a streamline, fluid elements are not permitted to cross each other. Finally, a relationship between the fluid elements in the spanwise direction is enforced. This relationship is commonly referred to as radial equilibrium as its derivation and frequent use is found in axial flow turbomachine codes. Radial equilibrium refers to the fact that the flow in a circular arc experiences a centripetal acceleration which must be equal to a net force in that direction which can only arise from the fluid pressure (Ref. 2). The form of the radial equilibrium equation changes depending on assumptions that can be made about the nature of the flowfield, but in its simplest form:

$$\frac{dp}{dr} = \rho \frac{V_\theta^2}{r} \tag{8}$$

Since the flow is not assumed to be exclusively in the axial direction it is more useful to relate the change in static pressure that occurs in the direction normal to the streamline. With a minor assumption that flow meridional angle is approximately constant then the incremental change in radius is equal to the incremental change in the spanwise direction times the cosine of the meridional angle:

$$\frac{dp}{ds} = \rho \frac{V_\theta^2 \cos\phi}{r} \tag{9}$$

Thus Equation (9) relates the change in static pressure occurring from one fluid element to the next. If one has n fluid elements whose condition at state 1 is known then there are $7n$ equations needed to fully determine all of the fluid elements' properties at state 2. The modified equation set is shown below with the required number of each equation preceding it for a total of $7n$:

$$n \qquad \dot{m}_{m2_i} = \dot{m}_{m1_i} \tag{1}$$

$$n \qquad h_{t2_i} - h_{t1_i} = \omega\left(r_{2_i} V_{\theta 2_i} - r_{1_i} V_{\theta 1_i}\right) \tag{2}$$

$$n \qquad p_{t2_i} = p_{t2\ \mathrm{ideal}_i} - f_{loss_i}(\mathrm{args}) \tag{3}$$

$$n \qquad \beta_{2_i} = \text{blade metal exit angle}_i + \delta_i \tag{4}$$

$$1 \qquad r_{2\,\mathrm{mean}} = r_{2\,\mathrm{machine}} \tag{5a}$$

$$n\text{-}1 \qquad r_{2\,\mathrm{inner}_{i+1}} = r_{2\,\mathrm{outer}_i} \tag{5b}$$

$$1 \qquad A_{m2\ \mathrm{mean}} = A_{m2\ \mathrm{machine}} \tag{6a}$$

$$n\text{-}1 \qquad \frac{dp}{ds_{2_i}} = \rho_{2_i} \frac{V_{\theta 2_i}^2}{r_{2_i}} \cos\phi_{2_i} \tag{6b}$$

$$n \qquad \phi_{2_i} = f_{\mathrm{machine}} \tag{7}$$

Note that these equations reduce to the meanline equations for the case where $n = 1$ and may therefore be used as the general set of equations for both meanline and streamline analyses.

The preceding set of equations reflect a direct-design approach where blade metal angles (Eq. (4)) and machine area (Eq. 6(a)) are input at design and the machine performance is the result. It is more common in conceptual design to use an inverse-design method where the desired machine performance is specified and the machine blade angles and area then result (Ref. 3). In theory this could be accommodated with an additional equation that relates the input blade metal exit angle to the blade angle that produces the desired design condition. In practice, the OTAC architecture uses flow angles as inputs with the blade angles calculated at design so Equation (4) can simply be replaced with a generic relationship Equation (4'):

$$n \qquad \text{design parameter} = \text{target value} \tag{4'}$$

where the design parameter may be pressure ratio across the blade segment, work extracted by the blade segment, or exit flow angle from the blade segment. The target value is a user-input radial distribution of the actual value of the variable specified by the design parameter. Similarly, Equation 6(a) can be replaced by other conditions which effectively set the exit area; for example, having meridional velocity remain constant through the blade row or specification of an exit axial Mach number both amount to setting the exit area. At off-design Equations (4) and (6a) take the form originally shown.

Using NPSS Objects to Solve the Desired Equations

The standard `FlowStation` object within NPSS represents a 1-D fluid state. The input attributes for the original `FlowStation` object were massflow rate, two state properties, and the flow velocity or Mach number. For OTAC, this object was modified such that it would calculate the 3-D velocity state and relative properties of a homogenous fluid. This modification requires an additional set of four input parameters, bringing the total to eight. These four added inputs were chosen to be flow meridional angle, flow tangential angle, flow radius, and rotational speed of the relative frame. Coupled with the original inputs, this allows calculation of each velocity component of interest and an upper and lower radial dimension that bounds the annular flow. Finally, the added rotational speed parameter permits calculation of fluid properties in a relative frame of reference which are useful quantities in the physics of rotating blade rows.

NPSS allows declaration of any number of instances of its `Solver` class. The NPSS `Solver` is a Newton-Raphson-based routine with robust features for user customization. Most importantly, the `Solver` class allows any number of custom independent variables and desired equations (called dependent conditions) to be added to its solution matrix. It was decided to make as much use of this as possible. First, each blade row has its own dedicated `Solver`. This means iterations to convergence for each blade row are handled by its `Solver` and each blade row solution is completed prior to moving to the next blade row in the machine. Furthermore, using the existing `Solver` object removes all coding responsible for solution iteration from the OTAC objects, leaving primarily the engineering and physics for the user and developer to see. This in turn makes it easier to understand what the code is doing and to modify it if necessary. Lastly, the ability to add custom independents and dependents means that OTAC objects can be formulated predominantly as a way to take a set of inputs and governing equations that define the given problem, eliminating the need to code the iterations and procedures to actually solve it.

Several other objects within the NPSS framework were also of use. The `FluidPort` object allows fluid information to be passed from object to object. Thus the fluid properties of a given streamline at the exit of a blade row are automatically transferred to the entrance of the following blade row. The `Shaft` Element is used as the means to input rotational speed of any blade row and its constituent segments. The `DataViewer` objects permit custom output to be sent a file.

Bladesegment Object

A diagram of the `BladeSegment` object is shown in Figure 3. The `BladeSegment` is an Element containing two of the modified `FlowStation` objects to track the incoming fluid state and the exiting fluid state. The incoming fluid state is known from data coming from the `FluidPort`, and the `BladeRow ShaftPort` provides the rotational speed if applicable. The outgoing fluid state needs to be determined. Since the rotational speed has already been input, each `BladeSegment` needs seven variables to set the fluid exit condition. Shown as the dashed arrow, seven independent variables set the exit state by fiat, but the values of these variables will be adjusted iteratively by the `Solver`. With this provisional exit state known, other calculations then proceed: the ideal exit state can be calculated to find the segment efficiency and “actual” loss parameter, the segment incidence angle and “actual” deviation are calculated, and empirical correlations from `Subelements` provide predictions of exit deviation angle and loss across the segment. Iterations are done by the `Solver` to ensure that the provisional exit state is driven toward a final, converged state such that the actual loss and deviation match the predicted loss and deviation in addition to satisfying all of the governing equations.

As with many turbomachinery programs, the user may specify an input variable set that is not physically possible, such as a large rise in pressure ratio associated with small turning or slow blade speed. In such cases the code will not converge to a solution and inform the user.

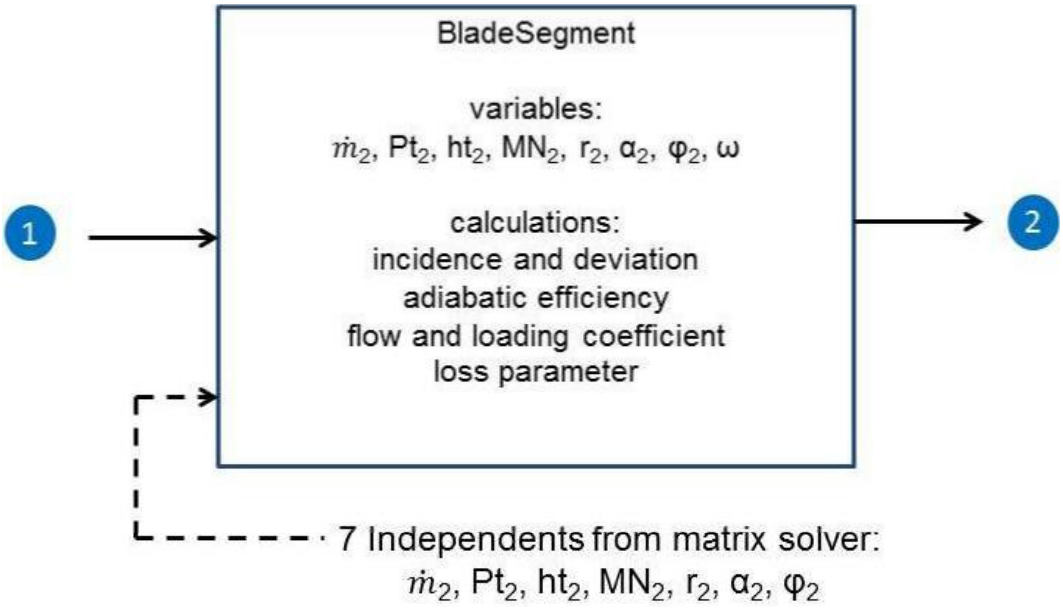

Figure 3.—Blade segment object schematic.

It is important to remain cognizant of the fact that the `BladeSegment` name can be misleading. The `BladeSegment` Element tracks the thermodynamic state changes and velocity state changes of an individual streamline as it passes through a segment of a blade. The `BladeSegment` does not represent a fixed, spanwise section of a blade: it represents a radially-varying segment of flow moving along a streamline through a blade row. Therefore its entrance and exit radii vary with the flow conditions.

Bladerow Object

The `BladeRow` object is schematically represented by Figure 4 for a five streamline example. The `BladeRow` is an `Assembly` containing a `BladeSegment`, `FluidInputPort`, and `FluidOutputPort` for each streamline desired by the user; a `ShaftPort` is also present if the `BladeRow` is rotating. While the behavior of each `BladeSegment` object must be independent of any other existing `BladeSegments` from an object-oriented programming perspective, the physical assumption of radial equilibrium requires that the solution for a particular `BladeSegment` depend upon the solution of other `BladeSegments`: the purpose of the `BladeRow` is to enable this.

Rather than create an onerous input burden for the user, it was decided to allow the user to simply input the number of streams desired and have each `BladeRow` object at runtime instantiate the appropriate number of `BladeSegments` within it; a consequence of this is that it also necessitated creation and instantiation of a set of `Solver Dependent` objects, the number of which varies depending on the number of streamlines. All of the `Independent` objects reside within the scope of the `BladeSegments` themselves: each `BladeSegment` contains its own seven `Independents` which for this example produces 35 `Independents` for the `BladeRow` as a whole. Of the 35 `Dependents`, some reside in the `BladeSegments` and the rest are contained in the `BladeRow` object. Five `Dependent` objects, representing Equations (1) to (4) and (7), reside in each `BladeSegment`. The `BladeRow` always has two `Dependents` representing Equations (5a) and (6a). This leaves eight `Dependents` to be created at runtime by the `BladeRow`; these `Dependents` are always 2*(*n*-1) in number and represent Equations (5b) and (6b).

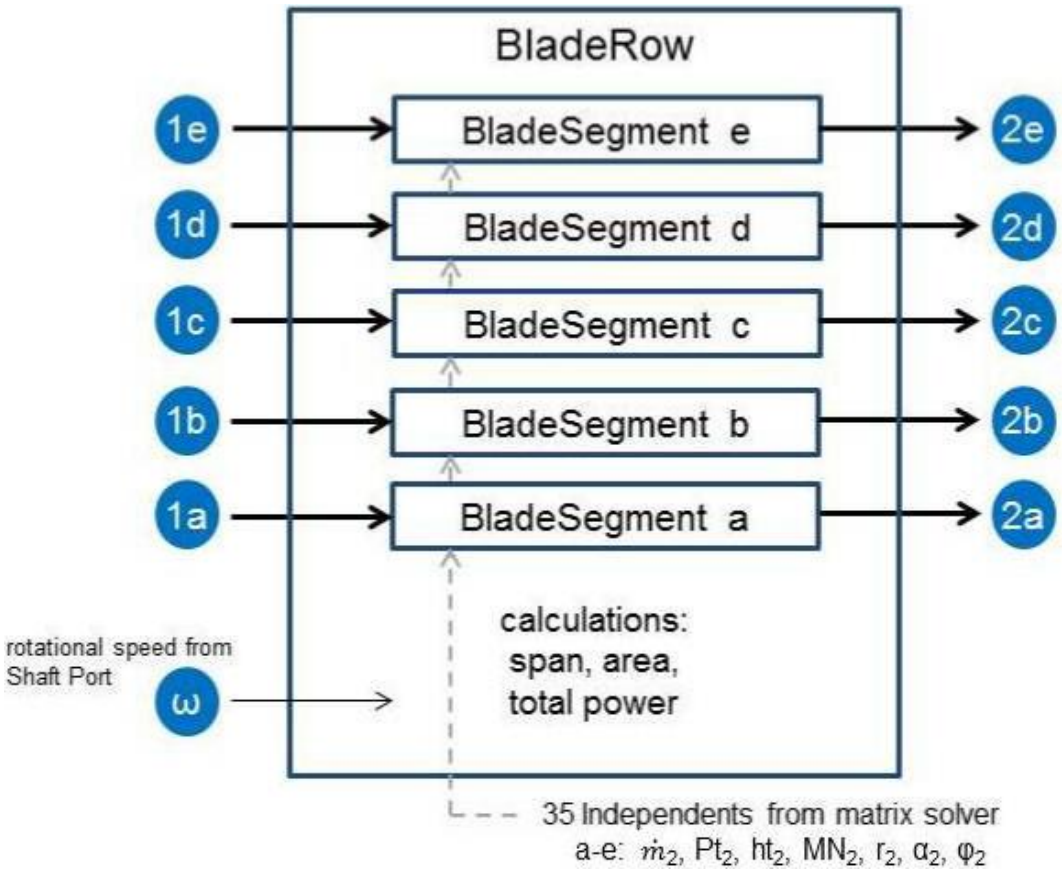

Figure 4.—BladeRow object schematic.

In addition to instantiation of all the necessary objects, the `BladeRow` performs some basic calculations given the state of the streamlines (i.e., `BladeSegments`) it contains such as determining mean radius and total power of the entire blade row. It must also estimate the pressure derivative used in the radial equilibrium equation; this is currently done with a simple finite difference between adjacent `BladeSegments`. The `BladeRow` also promotes each `BladeSegment` `FluidInputPort` and `FluidOutputPort` to the `Assembly` border so that `BladeRow` objects can be linked to other objects, including other `BladeRows`.

Since an inverse-design procedure is assumed, the `BladeRow` must also save the values of the blade metal angles determined at design. The values are used at off-design to determine by interpolation the entrance and exit metal angles associated with the changing flow radius of each `BladeSegment`.

Loss and Deviation Subelements

Loss mechanisms in turbomachinery cannot be analytically predicted; thus the designer uses empirical correlations to estimate such losses. OTAC uses `Subelement` objects located within the `BladeSegment` for this purpose. The `Subelement` returns a single value for the relative total pressure loss, $\bar{\omega}$, across the `BladeSegment` based on its internal calculations. The advantage of this approach is that loss models remain a separate entity allowing development of a suite of loss models applicable to specific kinds of loss mechanisms and specific types of turbomachinery. These loss models can be easily selected, switched out, modified, or stacked together by the user. The preceding discussion applies to deviation models as well, although they are typically not as complex as loss models. The deviation `Subelement` returns a value for δ based on its internal calculations.

Summary and Current Status

With the above objects it becomes possible to model any type of turbomachine. The user creates an input file with instances of `BladeRow` objects, each of which has its own attributes—rotating, non-rotating, design pressure ratio, loss model(s), etc. Based on the number of streams desired, OTAC will populate each `BladeRow` with the appropriate number of `BladeSegments` and set up the `Solver` objects for that `BladeRow`. The `Solver` then varies the exit state of each `BladeSegment` stream until the entire `BladeRow` solution has converged. The process then repeats for the next `BladeRow` until the entire machine design performance is completed. Operating characteristics may then be modified to run any number of off-design cases.

Currently an alpha release of OTAC has verified the capability of the code to perform its intended function. A beta release is planned for next year; the beta version will add default loss and deviation models, overall stage performance calculations, and additional elements to account for secondary flow addition/subtraction and radial mixing between blade rows.

References

1. Claus, Russel W., et al.: Numerical Propulsion System Simulation. Computing Systems in Engineering, vol. 2, no. 4, 1991, pp. 357–364.
2. Saravanamuttoo, Herb I.H., et al.: Gas Turbine Theory. Pearson Education Limited, Essex, England, 2009.
3. Japikse, David, and Baines, Nicholas C.: Introduction to Turbomachinery. Concepts ETI, Inc. and Oxford University Press, White River Junction, VT, 1997.